KneeHub_OKS Calibration Documentation Cleveland State University

Release 0.1

William Zaylor, Ammar Hafez, Jason P. Halloran

Sep 09, 2019

CONTENTS

CHAPTER

ONE

MODEL CALIBRATION SUMMARY

This section summarizes the steps involved in calibrating a finite element knee model. The summary includes links to other chapters where specific details are provided.

These steps apply to the OpenKnee(s) data set. The documentation for the DU data set can be found in a separate document.

1.1 Knee Model

The knee model that was developed during the *Model Development* phase is used for model calibration. Note that a simplified model will be used for calibration, where removal of various anatomy will increase computational efficiency. Details are described in the *Tibiofemoral Model* (page 4) section.

1.1.1 Coordinate Systems

Additional experimental data was made available for the OpenKnee(s) specimen, which will affect the coordinate frame placement from the Model Development phase. These data include the experimentally defined femoral and tibial coordinate systems, and the digitized registration markers. The registration markers are used to facilitate registration of the experimental femoral and tibial coordinate systems to the model's global coordinate system.

Specific documentation on the registration process can be found in the *registration* (page 4) section. The use of the experimental data found in the file `data-MC-oks003/State.cfg` to define the fixed tibial and "optimized" fixed femoral coordinate systems can be found in the *Tibia Coordinate System* (page 7) and *Femur Coordinate System* (page 8) sections. These sections include descriptions of how the data in `data-MC-oks003/State.cfg` was processed, including the conversion for a left knee.

1.2 Optimization Scheme

1.2.1 Optimization Test Cases

There is data for several laxity style test cases available across multiple flexion angles. A subset of these data is used in the calibration procedure. Each objective function evaluation will include 0° and 90° flexion and the following test cases: anterior drawer, posterior drawer, varus moment, valgus moment. The maximum loads from each test case will be included in the objective function, which will result in evaluation of 8 total loading points. See the *Test Cases* (page 16) and *Test Case Step* (page 25) sections for more information.

1.2.2 Control Variables

The knee model is composed of 11 ligament bundles, and the slack length for each bundle is a parameter in the optimization. Specifically, each ligament bundle has two slack length parameters, and these parameters are used to define the slack length of every fiber in the ligament bundle. More details can be found in the *Control Variables* (page 17) section.

Forward Kinematics Model Evaluation

A forward kinematic simulation was used to facilitate definition of the initial guess values as well as bounds for the control variables. The purpose of this simulation is to determine the maximum length of each ligament bundle throughout the experimental data used for calibration. The kinematics from the *calibration test cases* (page 16) are used as inputs to a kinematically driven finite element model, and the lengths of the fibers at the margins of each ligament bundle are recorded. More details can be found in the *Forward Kinematics Simulation* (page 19) section.

Control Variable Bounds

Upper and lower bounds for each control variable are used in the optimization. The lower bounds are set to prevent the optimization algorithm from evaluating slack lengths that may cause instability in the knee model. The upper bounds are used to limit the size of the parameter space. These bounds are defined using the values recorded from the *forward kinematics model evaluation* (page 19). The upper bound for each control variable is defined as the maximum length of the corresponding ligament fiber in the forward kinematics simulation results. The lower bounds are defined as 60% of the upper bound. More details can be found in the *Control Variable Bounds* (page 19) section.

Initial Guess

Each control variable's initial guess value is defined as 90% of the corresponding variable's upper bound (i.e. the maximum length throughout the kinematically controlled simulation). This approach is uniformly applied to every control variable in an attempt to avoid influencing the optimization solution with a judicious selection of initial guess values. More information can be found in the *Initial Guess* (page 19) section.

Optimization Algorithm

The gradient based "SLSQP" algorithm that is available in the SCIPY optimize toolbox will be utilized. This particular approach allows use of inequality constraints and bounds on the control variables. Details are described in *optimization scheme section* (page 16)

1.2.3 Objective Function

The objective function calculates the weighted sum of squared residual between model and experimentally measured kinematics during the simulated *test cases* (page 16). The objective function will evaluate the residual in anterior-posterior tibial displacement, varus-valgus rotation, and internal-external tibial rotation. More details can be found in the *Objective Function* (page 20) section. Information on the specific steps used in the Abaqus simulation can be found in the *Simulation Steps* (page 22) section.

1.2.4 Optimization Constraints

Inequality constraints are used to enforce that each *controlled ligament fiber* (page 17) experiences at least a nominal force of 0.1 N at one point through the simulated *calibration test cases* (page 16).

The constraints are used to prevent the optimization algorithm from effectively removing control variables from the model. A control variable can be effectively removed by specifying a large slack length where the ligament fibers are slack throughout the entire knee model evaluation. In this case the slack fibers will not influence the objective function, and this can result in the optimization algorithm “forgetting” the ligament when it otherwise should not. More details can be found in the *Constraints* (page 20) section.

CHAPTER

TWO

TIBIOFEMORAL MODEL

The tibiofemoral model that is used in the *optimization scheme* (page 16) is based on the model that was developed in the *Model Development* phase of this project. This section documents any changes that were made to the original model to facilitate the calibration procedure.

Given the model will likely be evaluated hundreds of times, specific anatomy were removed to increase computational efficiency. The patella is unloaded during the experimental laxity tests. As a result, the patella and the patellar ligaments were removed from the knee model. Additionally, the medial and lateral meniscus and mensical attachments were also removed. It is recognized that removing the meniscus may affect the estimate ligament slack lengths due to its potential to provide joint restraint. For the relatively low magnitudes of loading utilized in this work, it is assumed that minimal restraint is provided by the meniscus. This assumption will be tested during *confirmation of the calibration procedure* (page 16) and updated if necessary.

Additionally, the additional experimental data that was made available for the *Model Calibration* phase was used to define the *femoral* (page 8) and *tibial* (page 7) fixed coordinate systems. These new coordinate systems are used to develop the joint coordinate system using the same methods described in the *Model Development* documentation. The difference in this model being the fixed femoral and tibial coordinate systems are defined using experimental data instead of manually digitized landmarks from the MR images.

The following sections offer more detail on the model that is used in the *optimization scheme* (page 16).

2.1 Ligament Properties

Each ligament bundle's material properties include two slack length values, toe region percent, and stiffness. The calibration procedure will target bundle-specific *slack lengths* (page 17). The toe region percent and the stiffness values are not included in the calibration procedure and will be set to values defined during *Model Development*. In short, ligament stiffness values are defined based on literature values and all toe regions are be set at 6% strain.

2.1.1 Registration

The purpose of the registration is to place consistent bone-specific coordinate frames between the experiment and model. In practice, the transform (T_{Im_Sen}) will be used to define the position and orientation of the femur and tibia's position sensor in the MR image coordinate system. Note that in the experiment, the femoral and tibial embedded coordinate systems are defined with respect to the corresponding body's position sensor (the IRED sensors used to track bony motion). Registration places the fixed tibia and femoral coordinate systems within the MR image's coordinate system, which is effectively defined using the centers of registration spheres that are common to both the experiment and MR images.

Fiducial (or "registration") spheres that are fixed to the femur and tibia are used to facilitate the registration process. The sphere surfaces were digitized during experimental setup and reported in the `data-MC-oks003/State.cfg` file. See *Digitized Points - Experimental* (page 5) for details on how the reported experimental data was used to define the digitized points with respect to the appropriate body's position sensor. The spheres are also visible in the MR images. 3D Slicer is used to manually place 30 points around the perimeter of each fiducial to "digitize" the fiducials in the MR image's coordinate system.

The digitized points are used in a sphere fitting procedure to estimate the center of each fiducial (see https://jekel.me/2015/Least-Squares-Sphere-Fit/ for more details). The points that define the center of each sphere, with respect to the image and position sensor's coordinate system, are used as inputs to a landmark registration algorithm (Fig. 2.1). The algorithm computes the transform (T_{Im_Sen}) that gives the least squared fit between the given target points and the transformed source points (https://vtk.org/doc/nightly/html/classvtkLandmarkTransform.html). In this application, the fiducial centers in the MR image's coordinate system are the target points, and the centers in the rigid body's sensor coordinate system are the source points.

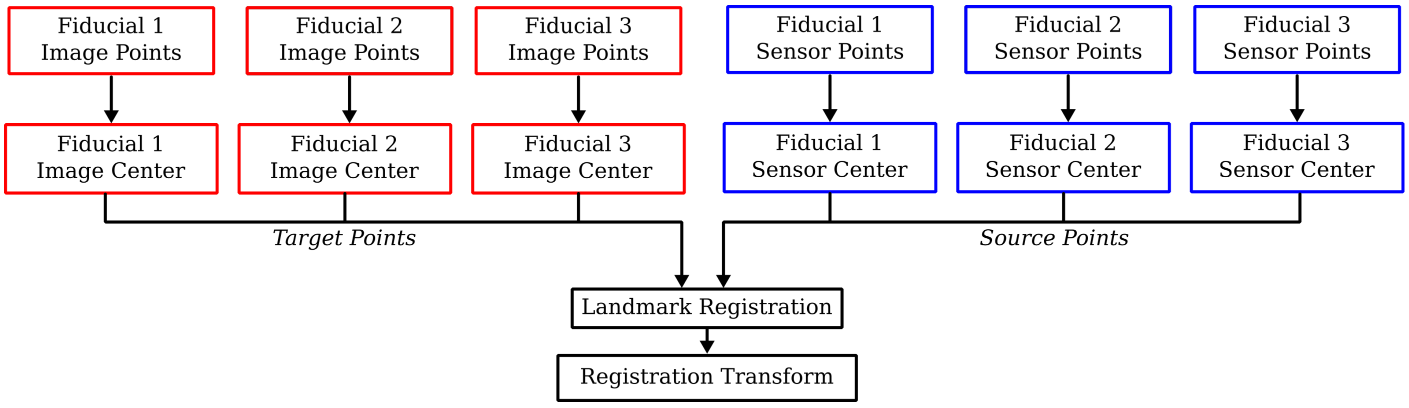


Fig. 2.1: A flowchart demonstrating how the digitized fiducial points in the MR image and the position sensor's coordinate system are used to define the transform used for registration for a rigid body, such as the femur or tibia.

Digitized Points - Experimental

This section describes how the digitized point data that are reported in the `data-MC-oks003/State.cfg` file are used to define the points relative to the corresponding body's position sensor. This demonstrates an example for one point on rigid body 1, however the same process is repeated for the remaining 9 points on rigid body 1, and the same process also applies to rigid body 2.

Below you can see the relevant data from that file (note that only one digitized point is shown):

```
[MRI Fiducial Sphere Positions]
... (data not shown)
Collected Points Rigid Body 1 (m) = "<size(s)=30 3> 0.9900980530 0.7572836873 0.
↪0611828908
Collected Points Rigid Body 1 Position Sensor (m,rad) = 1.0580799235 0.8044003377 0.
↪0868033719 1.8617229826 0.0655618814 1.0999982884
```

In the above example, the desired variables are

$$\begin{aligned} P_{Wx} &= 0.9900980530 \\ P_{Wy} &= 0.7572836873 \\ P_{Wz} &= 0.0611828908 \\ t_x &= 1.0580799235 \\ t_y &= 0.8044003377 \\ t_z &= 0.0868033719 \\ \alpha &= 1.8617229826 \\ \beta &= 0.0655618814 \\ \gamma &= 1.0999982884 \end{aligned}$$

The line labeled `Collected Points Rigid Body 1 (m)` is used to define the coordinates of the digitized points relative to the Optotrak world coordinate system. Each set of three values defines the x, y, z coordinates of P_W, which is the digitized point relative to the Optotrak world. These values are converted from meters to millimeters.

The line labeled `Collected Points Rigid Body 1 Position Sensor (m,rad)` is used to define the position of rigid body 1's position sensor relative to the Optotrak world coordinate system. These six values define the translations t_x, t_y, t_z and rotations α, β, γ of rigid body 1's position sensor. The translations are converted from meters to millimeters.

The values $[t_x, t_y, t_z, \alpha, \beta, \gamma]$ are used to define a transformation matrix T_{W_R1} which is the transform from the Optotrak world coordinate system to the position sensor's coordinate system. The transofrm matrix (T_{W_R1}) is defined as

$$T_{W\_R1} = r_z r_y r_x$$

Where

$$r_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\alpha) & -sin(\alpha) & 0 \\ 0 & sin(\alpha) & cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$r_y = \begin{bmatrix} cos(\beta) & 0 & sin(\beta) & 0 \\ 0 & 1 & 0 & 0 \\ -sin(\beta) & 0 & cos(\beta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$r_z = \begin{bmatrix} cos(\gamma) & -sin(\gamma) & 0 & 0 \\ sin(\gamma) & cos(\gamma) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and the translations (t_i) are then specified as the last column in T_{W_R1} as

$$T_{0,3} = t_x$$

$$T_{1,3} = t_y$$

$$T_{2,3} = t_z$$

The transformation matrix (T_{W_R1}) is used to define a point that is recorded relative to the Optotrak world (P_W) and specify its coordinates relative to rigid body 1's position sensor (P_{R1})

$$P_{R1} = T^{-1}_{W\_R1} P_W$$

Note: To avoid error propagation, a python function from numpy (`P_{R1} = numpy.linalg.solve(T_{W\_R1}, P_W)`) is used to solve the system of linear equations for P_{R1}. This is used to avoid inverting the transformation matrix T_{W_R1}. Also note that to ensure that the dimensions are compatible, 1. is appended to the end of P_W to create an array with a shape of 1x4. The first three values in P_{R1} are taken as the x, y, z coordinates.

2.1.2 Tibial Origin Orientation

The orientation of the node that is coincident with the fixed tibial coordinate system is defined. This facilitates load application with respect to the tibia's fixed coordinate system.

Transform

Three points are used to define the transform that is used to orient the origin of the fixed tibial coordinate system. One point is the origin of the fixed tibial coordinate system, and the other two are coincident with the x and y axis of the tibia's fixed coordinate system. The point at the origin has already been defined when generating the joint coordinate system (see *Model Development* documentation for more information on the joint coordinate system's connector elements). The other two points are defined with the equation below.

$$p_i = p_{origin} + n_i$$

Where p_i is the node along the i^{th} axis of the fixed tibial coordinate system, n_i is the corresponding unit vector that is parallel to the i^{th} axis of the fixed tibial coordinate system, and p_{origin} is the origin of the fixed tibial coordinate system.

In Abaqus, the `*Transform` option is used to define the transform, and this takes six values. The first three values are the difference between the coordinates of the origin and the point along the tibia's x-axis ($p_x - p_{origin}$). Similarly, the second set of three values is the difference between the coordinates of the origin and the point along the tibia's y-axis ($p_y - p_{origin}$).

2.1.3 Tibia Coordinate System

The tibia's fixed coordinate system is defined using data reported during experimental setup and the transform (T_{Im_Sen}) defined during the registration for the tibia (*Registration* (page 4)). The file `data-MC-oks003/State.cfg` contains the transform that is used to define the tibia's fixed coordinate system with respect to the tibia's position sensor. Below you can see the relevant data from `data-MC-oks003/State.cfg` with unused data removed:

```
[Knee JCS]
T_Sensor2_RB2 = "<size(s)=4 4> 0.5805398524 0.7813549675 0.2290368848 0.0748743152 0.
↪0563481400 0.2420640867 -0.9686226639 -0.0471447919 -0.8122797344 0.5752298608 0.
↪0964999501 -0.0152677242 0.0000000000 0.0000000000 0.0000000000 1.0000000000"
```

Where `T_Sensor2_RB2` is converted to a 4x4 matrix. The first three rows in the last last column of `T_Sensor2_RB2` are the translations for the transform, and these values are multiplied by 1000 to convert from meters to millimeters.

$$T_{Sensor2\_RB2} = \begin{bmatrix} 0.58053985 & 0.78135497 & 0.22903688 & 74.8743152 \\ 0.05634814 & 0.24206409 & -0.96862266 & -47.1447919 \\ -0.81227973 & 0.57522986 & 0.09649995 & -15.2677242 \\ 0. & 0. & 0. & 1. \end{bmatrix}$$

Given that oks003 is a left knee specimen, $T_{Sensor2\_RB2}$ is reflected about the x-axis, which is shown below. The reflection of $T_{Sensor2\_RB2}$ is $R_{Sensor2\_RB2}$ (note that this is not needed for a right knee specimen).

$$R_{Sensor2\_RB2} = T_{Sensor2\_RB2} * \begin{bmatrix} 1 & -1 & -1 & -1 \\ -1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0.58053985 & -0.78135497 & -0.22903688 & -74.8743152 \\ -0.05634814 & 0.24206409 & -0.96862266 & -47.1447919 \\ 0.81227973 & 0.57522986 & 0.09649995 & -15.2677242 \\ 0. & 0. & 0. & 1. \end{bmatrix}$$

Next the transform from the MR image's global coordinate system to the tibia's fixed coordinate system in the MR image (T_{Im_RB2}) is defined. T_{Im_Sen2} (which is determined from *registration* (page 4) of rigid body 2 fiducial points) is multiplied by $R_{Sensor2\_RB2}$

$$T_{Im\_RB2} = T_{Im\_Sen2} R_{Sensor2\_RB2}$$

T_{Im_RB2} can be used to define the tibia's fixed coordinate system in the knee model using the SimVitro convention, where the positive direction anterior-posterior axis points posteriorly. To match the convention used by this lab, the fixed tibial coordinate system defined by T_{Im_RB2} is rotated 180 degrees about the fixed tibia's z-axis. This defines the positive direction along the anterior-posterior axis as pointing anteriorly.

$$S_{Im\_RB2} = T_{Im\_RB2} * \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The last row in S_{Im_RB2} is neglected, and the columns are used to define the axes of the fixed tibial coordinate system and its origin.

$$\begin{aligned} x_{axis} &= [S_{Im\_RB2}(0,0), S_{Im\_RB2}(1,0), S_{Im\_RB2}(2,0)] \\ y_{axis} &= [S_{Im\_RB2}(0,1), S_{Im\_RB2}(1,1), S_{Im\_RB2}(2,1)] \\ z_{axis} &= [S_{Im\_RB2}(0,2), S_{Im\_RB2}(1,2), S_{Im\_RB2}(2,2)] \\ origin &= [S_{Im\_RB2}(0,3), S_{Im\_RB2}(1,3), S_{Im\_RB2}(2,3)] \end{aligned}$$

2.1.4 Femur Coordinate System

The femur's fixed coordinate system is defined using data reported during experimental setup and the transform (T_{Im_Sen}) defined during the registration for the femur (*Registration* (page 4)). The file `data-MC-oks003/State.cfg` contains the transforms that are used to define the femur's fixed coordinate system with respect to the femur's position sensor.

Note: The "optimized" femoral coordinate system is used, so additional experimental data needs to be parsed from the file `data-MC-oks003/State.cfg`

Below you can see the relevant data from `data-MC-oks003/State.cfg` with unused data removed:

```
[JCS]
T_FEMnew_FEMold = "<size(s)=4 4> 0.9981824400 0.0492132086 -0.0347832805 0.0009999990␣
↪-0.0492330265 0.9987872780 0.0002870382 -0.0119875025 0.0347552241 0.0014259697 0.
↪9993948374 -0.0007254170 0.0000000000 0.0000000000 0.0000000000 1.0000000000"
```

(continues on next page)

(continued from previous page)

```
[Knee JCS]
T_Sensor1_RB1 = "<size(s)=4 4> -0.7583937513 -0.6495517799 0.0540500060 0.0316151825 -
↪0.1168279952 0.2170479726 0.9691446730 -0.1143759355 -0.6412410915 0.7286787103 -0.
↪2404936584 -0.0050802701 0.0000000000 0.0000000000 0.0000000000 1.0000000000"
```

Where `T_FEMnew_FEMold` and `T_Sensor1_RB1` are each converted to a 4x4 matrix. The first three rows in the last last column of `T_FEMnew_FEMold` and `T_Sensor1_RB1` are the translations for the transforms, and these values are multiplied by 1000 to convert from meters to millimeters.

$$T_{FEMnew\_FEMold} = \begin{bmatrix} 0.998182440 & 0.0492132086 & -0.0347832805 & 0.999999000 \\ -0.0492330265 & 0.998787278 & 0.000287038200 & -11.9875025 \\ 0.0347552241 & 0.00142596970 & 0.999394837 & -0.725417000 \\ 0. & 0. & 0. & 1. \end{bmatrix}$$

$$T_{Sensor1\_RB1} = \begin{bmatrix} -0.75839375 & -0.64955178 & 0.05405001 & 31.6151825 \\ -0.116828 & 0.21704797 & 0.96914467 & -114.3759355 \\ -0.64124109 & 0.72867871 & -0.24049366 & -5.0802701 \\ 0. & 0. & 0. & 1. \end{bmatrix}$$

The transform from femur's position sensor to the optimized femoral coordinate system ($T_{Sensor1\_FEMnew}$) is defined as

$$T_{Sensor1\_FEMnew} = T_{Sensor1\_RB1} T_{FEMnew\_FEMold}^{-1}$$

Given that oks003 is a left knee specimen, $T_{Sensor1\_FEMnew}$ is reflected about the x-axis, which is shown below. The reflection of $T_{Sensor1\_FEMnew}$ is $R_{Sensor1\_FEMnew}$ (note that this is not needed for a right knee specimen).

$$R_{Sensor1\_FEMnew} = T_{Sensor1\_FEMnew} * \begin{bmatrix} 1 & -1 & -1 & -1 \\ -1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Note: Array multiplication is used in the above equation, so element-wise multiplication is used (i.e. there is no summing over indicies).

Next the transform from the MR image's global coordinate system to the femur's optimized fixed coordinate system in the MR image (T_{Im_Sen1}) is defined. T_{Im_Sen1} (which is determined from *registration* (page 4) of rigid body 1 fiducial points) is multiplied by $R_{Sensor1\_FEMnew}$

$$T_{Im\_FEMnew} = T_{Im\_Sen1} R_{Sensor1\_FEMnew}$$

T_{Im_FEMnew} can be used to define the femur's optimized fixed coordinate system in the knee model using the SimVitro convention, where the positive direction anterior-posterior axis points posteriorly. To match the convention used by this lab, the fixed optimized femoral coordinate system defined by T_{Im_FEMnew} is rotated 180 degrees about the fixed optimized femur's z-axis. This defines the positive direction along the anterior-posterior axis as pointing anteriorly.

$$S_{Im\_FEMnew} = T_{Im\_FEMnew} * \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The last row in S_{Im_FEMnew} is neglected, and the columns are used to define the axes of the fixed optimized femoral coordinate system and its origin.

$$\begin{aligned}
x_{axis} &= [S_{Im\_FEMnew}(0,0), S_{Im\_FEMnew}(1,0), S_{Im\_FEMnew}(2,0)] \\
y_{axis} &= [S_{Im\_FEMnew}(0,1), S_{Im\_FEMnew}(1,1), S_{Im\_FEMnew}(2,1)] \\
z_{axis} &= [S_{Im\_FEMnew}(0,2), S_{Im\_FEMnew}(1,2), S_{Im\_FEMnew}(2,2)] \\
origin &= [S_{Im\_FEMnew}(0,3), S_{Im\_FEMnew}(1,3), S_{Im\_FEMnew}(2,3)]
\end{aligned}$$

CHAPTER

THREE

EXPERIMENTAL DATA

Placeholder

3.1 Extract Experimental Data

The OpenKnee(s) experimentally measured kinematics and kinetics are reported in .tdms files. These files contain three sets of data that are used in model calibration.

1) `Kinetics.JCS.Desired`, The kinetics that were specified for the test case.
2) `State.JCS Load` , The joint kinetics that are measured during the test case. These loads are taken to be reported with respect to the fixed tibial coordinate system. The following is from the OpenKnee(s) documentation `This is the kinetic state of the joint (i.e. tibia loads). More specifically, it is the joint (tibia) loads that are ...` This can be found on page 8 of this document: https://simtk.org/plugins/moinmoin/openknee/Infrastructure/ExperimentationMechanics?action=AttachFile&do=get&target=2013CB-031-002.B+simVITRO+Data+File+Contents_Open+Knee.pdf
3) `State.JCS`, The measured joint kinematics derived from the robot position and defined using the *optimized femur* coordinate system.

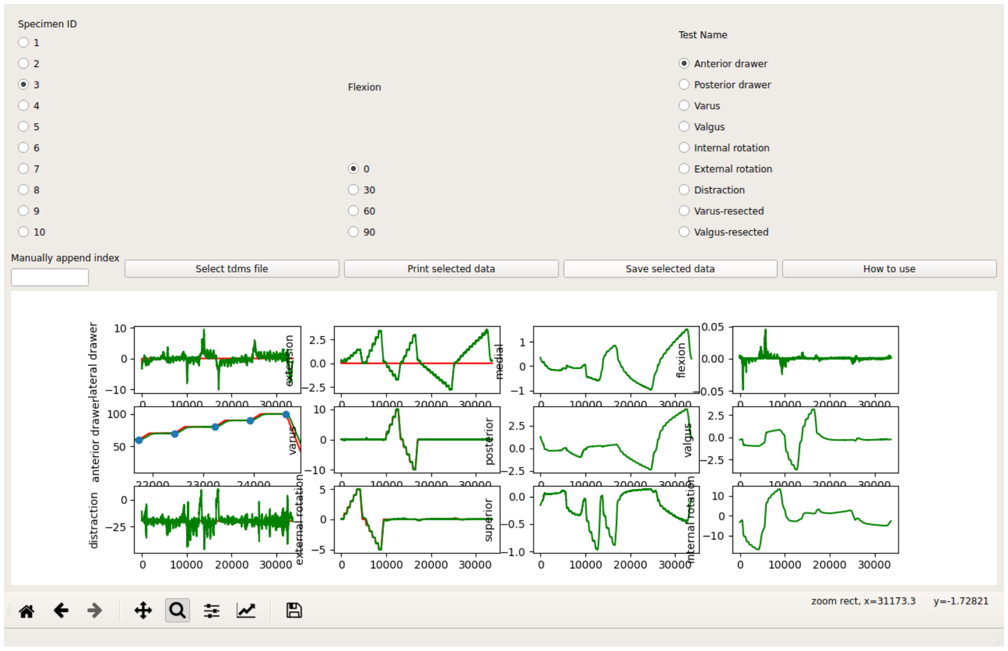
These data are time-synced, and it is assumed that the values at each given index correspond to the other reported values at the same index. During preprocessing for model calibration, the indicies of the desired points in the experimental test case are defined, and these indicies are later used to extract specific kinematics and kinetics from the .tdms file.

The specified (`Kinetics.JCS.Desired`) kinetics are used to parse the experimental data. The specified data ramps the load up by a certain amount, then holds the load for a time before ramping to the next specified load. A custom program (Fig. 3.1) is used to select the data at the end of a "hold" step (Fig. 3.2). This program is used to plot the specified kinetics for the load cases that are reported in the .tdms file. The user then selects the range that is desired, and custom python scripts are used to select the points at the end of each ramp's "hold" step (Fig. 3.2). The indicies of the selected points are recorded in a .xml file, along with metadata, including the specimen number, name assigned to the test case, and the .tdms file that was used to extract the data. The recorded indicies are later used to extract the experimentally joint kinematics and kinetics from the .tdms file.

3.2 Process Experimental Data - OpenKnee(s)

3.2.1 Kinematics

The experimentally measured tibial kinetics are imported from the appropriate `.tdms` file, and the data under the heading `State.JCS` are utilized. Two adjustments are made to the experimental data. The first converts the relative joint kinematics reported in `State.JCS` to absolute joint kinematics. The second converts to the sign conventions used by CSU.



Specimen ID
1
2
[✓ 3]
4
5
6
7
8
9
10

Flexion
[✓ 0]
30
60
90

Test Name
[✓ Anterior drawer]
Posterior drawer
Varus
Valgus
Internal rotation
External rotation
Distraction
Varus-resected
Valgus-resected

Manually append index

zoom rect, x=31173.3 y=-1.72821

Fig. 3.1: The GUI of the the semi-automatic process of extracting experimental data.

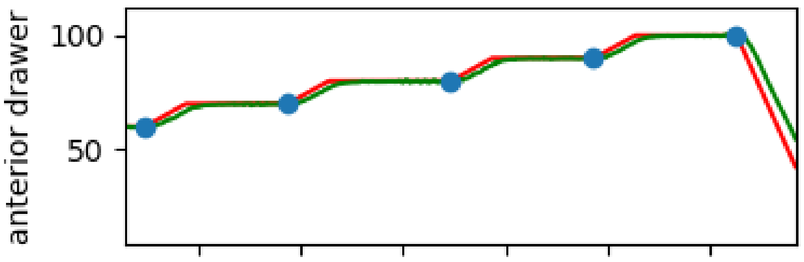


Fig. 3.2: A larger view of the last five selected steps in the anterior drawer test at 0° flexion. Note the red line shows the specified joint kinetics (`Kinetics.JCS.Desired`), and the green line shows the experimentally measured kinetics (`State.JCS Load`).

Kinematics Adjustment 1 - Offset

The experimentally measured kinematics from OpenKnee(s) reports the change in kinematics relative to a "neutral" joint position. These kinematics are adjusted to the absolute joint kinematics because the knee model inputs are absolute kinematics, and the model's kinematic results are reported as absolute kinematics. This adjustment is made by applying a constant offset to the reported experimental kinematics. This offset is the joint kinematics necessary needed to achieve a experimentally measured "neutral" position.

The kinematics needed to achieve this "neutral" joint position are reported in the `State.cfg` file. Given that the kinematics from the `.tdms` file under the `State.JCS` heading are utilized, the neutral position from the `[JCS]` block of the `State.cfg` file is used. Below is an example of the relevant parts from the `State.cfg` file.

```
[JCS]
Position Offset (m,rad) = "<size(s)=6> -0.0006018416 -0.0042514883 -0.0255631685 0.
↪1450310182 0.0232244115 -0.2681618244"
```

These values are parsed using custom Python scripts, and the values are converted from m to mm, and radians to degrees. These values are taken to correspond to the following kinematics in this order: [Medial tibial translation, Posterior tibial translation, Superior tibial translation, Flexion, Valgus, Internal tibial rotation].

The absolute joint kinematics are defined by adding the offset values to the corresponding kinematics from the .tdms file (`State.JCS`).

Kinematics Adjustment 2 - CSU convention

The joint kinematics are converted to the convention used by CSU after the *offsets are applied* (page 13) to the kinematics. This adjustment is made to keep the experimental descriptions of motion consistent the the descriptions of motion output by the knee model. The CSU convention describes motions as:

- Medial tibial translation
- Anterior tibial translation
- Superior tibial translation
- Flexion
- Varus
- Internal tibial rotation.

To convert to the CSU convention, the sign of the kinematics for two motions are changed:

1) `JCS Posterior` kinematics are multiplied by -1 to convert to anterior tibial translation.
2) `JCS Valgus` kinematics are multiplied by -1 to convert to varus rotation.

3.2.2 Kinetics

The experimentally measured tibial kinetics are imported from the appropriate `.tdms` file, and the data under the heading `State.JCS Load` are used to define the load inputs for the *test case simulation step* (page 25). However, the data is processed to adjust the description of the positive loading directions. These adjustments are made by changing the sign of the reported data.

The following sections described how the experimentally measured loads are adjusted. Note that the sign changes may be redundant, however this is done intentionally to keep a consistent workflow between data sources and between right and left knee models.

Kinetics Adjustment 1 - CSU convention

The first adjustment is used to change from the reported SimVitro convention on directions to the CSU convention. The CSU convention has the positive x direction pointing to the right, and the positive y direction pointing anteriorly. The signs of the data reported in the .tdms file under `State.JCS Load` are adjusted based on their descriptions (Table 3.1). This step is not strictly necessary, however this keeps the data consistent with the modeling workflow used at CSU.

These adjustments are made when the data is imported from the `.tdms` file, and before the loading profiles are defined for the Abaqus simulation. The loading profiles are defined using these adjustments, and the data can be found in the appropriate `.inp` files. Below is a description of the adjustments described in Table 3.1

- `Lateral Drawer` is converted to medial tibial drawer by multiplying the corresponding experimental data by -1
- `Anterior Drawer` is not adjusted
- `Distraction` is not adjusted
- `Extension Torque` is converted to flexion torque by multiplying the corresponding experimental data by -1
- `Varus Torque` is not adjusted
- `External Rotation Torque` is converted to internal tibial rotation torque by multiplying the corresponding experimental data by -1

Kinetics Adjustment 2 - Right or Left Knee

The signs of the loads may be changed depending on whether a right or left knee specimen is being simulated. This is because the loads are applied with respect to the fixed tibial coordinate system, and the positive directions of this coordinate system may not match the experimental data's loading directions.

For example, based on the CSU convention, the positive x direction points to the right. The SimVitro reported load that corresponds to this direction is `Lateral Drawer`. For a right knee, the `Lateral Drawer` direction and the tibia's x-axis point in the same direction. Conversely, for a left knee the `Lateral Drawer` direction and the tibia's x-axis point in opposite directions. The sign of the reported data is changed because the loads are applied with respect to the tibia's fixed coordinate system.

These adjustments are applied in the part of the Abaqus `.inp` file that is used to define the *test cases* (page 25) step. An example of these adjustments can be seen in the *code block* (page 15) below.

Below is a description of how the experimental loads are adjusted for each degree of freedom.

- After the previous adjustment (Table 3.1), the data is described as `medial tibial drawer`. For a right knee, the data is multiplied by -1 (Table 3.1) to change the load to `lateral tibial drawer`.
- The data is reported as `anterior tibial drawer`, and not adjusted in the previous step (Table 3.1). The anterior direction is in the positive y relative to the fixed tibial coordinate system for left and right knees, so this value is not changed for right or left knees (Table 3.1).
- The data is reported as `distraction`, and not adjusted in the previous step (Table 3.1). A load that acts in the inferior direction relative to the fixed tibial coordinate system will cause joint distraction regardless of a right or left knee. The positive direction for the tibia's z-axis points in the superior direction, therefore the sign for both right and left knees is changed to make the distraction load act in the inferior direction (Table 3.1).
- The flexion angle of the joint is prescribed, so the extension torque is not applicable.
- The data is reported as `varus torque`, and not adjusted in the previous step (Table 3.1). The y-axis of tibia's fixed coordinate system points in the anterior direction. For a right knee, a positive torque around

the tibia's y-axis causes varus, so the sign is not changed for a right knee. However the sign is changed for a left knee because a positive torque about the y-axis of a right tibia causes valgus.

- After the previous adjustment (Table 3.1), the data is described as `internal tibial rotation torque`. A positive rotation about the positive z-direction of a right knee causes internal tibial rotation, so the sign is not changed for a right knee. However the sign is changed for a left knee because a positive rotation about the positive z-direction of a left knee causes external tibial rotation.

Table 3.1: The degree of freedom at the node that the loads are applied to, and the corresponding load description, and the relevant sign convention changes. The reported description is from the given .tdms files that contain experimental data under the `State.JCS Load` heading.

Axis	Reported	CSU convention	Right knee	Left knee
x_1	JCS Load Lateral Drawer_LP Filtered 1.0 Hz	-1	-1	1
x_2	JCS Load Anterior Drawer_LP Filtered 1.0 Hz	1	1	1
x_3	JCS Load Distraction_LP Filtered 1.0 Hz	1	-1	-1
x_4 (N/A)	JCS Load Extension Torque_LP Filtered 1.0 Hz	-1	1 (N/A)	1 (N/A)
x_5	JCS Load Varus Torque_LP Filtered 1.0 Hz	1	1	-1
x_6	JCS Load External Rotation Torque_LP Filtered 1.0 Hz	-1	1	-1

Below is an example code showing how loads are specified in the Abaqus input file for a simulation of a left knee:

```
** Medial tibial drawer force, -1. for right knee and 1. for left knee
*CLOAD, amplitude=medialTibialDrawerForce, follower, op=new
JointCoordSys.1, 1, 1.
** Anterior tibial drawer force, 1. for right and left knee
*CLOAD, amplitude=anteriorTibialDrawerForce, follower, op=new
JointCoordSys.1, 2, 1.
** Distraction force, -1. for right and left knee
*CLOAD, amplitude=distractionForce, follower, op=new
JointCoordSys.1, 3, -1.
** Varus torque, 1. for right knee and -1. for left knee
*CLOAD, amplitude=varusTorque, follower, op=new
JointCoordSys.1, 5, -1.
** Internal tibial rotation torque, 1. for right knee and -1. for left knee
*CLOAD, amplitude=internalTibialRotationTorque, follower, op=new
JointCoordSys.1, 6, -1.
```

Where each amplitude is the magnitude of the corresponding load, where the adjustments from *adjustment 1* (page 14) are reflected.

The node `JointCoordSys.1` is the node that is coincident with the origin of the tibia's fixed coordinate system. The integer following `JointCoordSys.1` indicates the degree of freedom that the load is applied to, and this corresponds with x_i in Table 3.1. The final integer is multiplied by the specified amplitude. This is used to adjust for a right or left knee according to Table 3.1.

CHAPTER

FOUR

OPTIMIZATION SCHEME

A sequential least squares programming (SLSQP) optimization algorithm will be used to calibrate ligament properties. This algorithm was selected because it allows for *inequality constraints* (page 20) and *control variable bounds* (page 19). This algorighm is implimented in the Scipy optimize module.

Initially the convergence tolerance will be set to 1.e-4. It is anticipated this value should be sufficiently small for these analyses, however that will need to be determined at the time of running the calibration procedure. The analyst may observe the solutions have essentially converged, but have not reached the assigned tolerance. To prevent an inordinately long calibration process, the optimization may be terminated prematurely and the lowest objective value used as the calibrated results. If the tolerance value is updated and/or early termination is utilized it will be documented as a deviation.

The SLSQP algorithm is gradient based, and the finite difference method is used to define the gradient. A difference step of -0.1 mm is used in the finite difference calculation. A negative value is used to address cases where a control variable is near a *constraint* (page 20). When a control variable is near a constraint, the corresponding ligament fiber experiences little load during the simulated *test cases* (page 16). Using a negative difference step decreases the ligament fiber's slack length and likely increases that fiber's load during the simulated *test cases* (page 16). This is an attempt to avoid violating *constraints* (page 20) in the finite difference calculation. Note that this finite difference calculation is executed using in-house python code, which allows for direct control of the difference step, and also allows for parallelization of the gradient evaluation.

The following sections offer more details on the parameters used to setup and run the optimization.

4.1 Test Cases

The individual experimental laxity tests that were performed are referred to as a "test case". Not all of the test cases were used for model calibration. This section describes which test cases were used for model calibration, and which test cases were used to evaluate the results of the model (referred to as "Confirmation" test cases).

4.1.1 Calibration Test Cases

The calibration load cases are the the experimental tests that are simulated as part of the *objective function* (page 20) evaluation. These load cases are a subset of the available data. The *objective function* (page 20) includes the maximum applied loads during the anterior-posterior drawer and varus-valgus tests at $0°$ and $90°$ flexion.

4.1.2 Confirmation Test Cases

The experimental test cases that were excluded from the optimization are used to evaluate the estimated ligament properties. These tests include the anterior-posterior drawer and varus-valgus tests at $30°$ and $60°$ flexion, and the internal-external rotation tests at $0°$, $30°$, $60°$ and $90°$ flexion. This evaluation will use the calibrated model to simulate these test cases and measure the error between the calibrated model and experimentally measured kinematics.

Note: Additional test cases and/or test points will be added to the set of calibration test cases if (1) the optimization does not converge or (2) there is excessive error between the calibrated model and experimentally measured kinematics for the calibration and confirmation test cases. Excessive error is defined as kinematic RMS errors that are equal to or greater than $4°$ for rotations and 4 mm for translations, as would be roughly consistent with recent modeling efforts [EKH+15] and [HCA+16].

4.2 Control Variables

There are two control variables that specify the slack length for each ligament bundle. One control variable specifies the slack length of one margin fiber, and the other specifies the slack length of the other margin fiber (Fig. 4.1). The given slack lengths are used to calculate the strain at the margin fibers of the ligament bundle, and these strains are linearly interpolated to define the slack lengths of the remaining 23 fibers in the ligament bundle.

Note: A ligament bundle's two given slack lengths are not directly interpolated to assign slack length to the remaining 23 fibers. This is because the femoral and tibial fiber insertion points likely do not lie in the same plane. Additionally, the imaged *insertion-to-insertion length* (page 17) may vary across the ligament bundle due to wrapping.

The given slack lengths for a ligament bundle's margin fibers, x_0 and x_1 and the fiber's imaged *insertion-to-insertion length* (page 17), L_0^{im} and L_1^{im}, are used to calculate the strain (ϵ^*) needed to be applied to the fiber's imaged insertion-to-insertion length to achieve the given slack lengths (x_0, x_1).

$$\epsilon_i^* = \frac{x_i - L_i^{im}}{L_i^{im}} \tag{4.1}$$

Note: ϵ_i^* is the strain that is applied to the ligament relative to the fibers images *insertion-to-insertion length* (page 17). This is not the prestrain carried by the fiber, which is relative to the slack length, $\epsilon = \frac{L^{im} - L^{slack}}{L^{slack}}$

The strains for the two margin fibers (ϵ_0^* , ϵ_1^*) are linearly interpolated across the ligament bundle. Each fiber's slack length is calculated using the interpolated strain value and imaged *insertion-to-insertion length* (page 17) (4.1).

4.2.1 Insertion-to-Insertion Length

The insertion-to-insertion length of a ligament fiber is defined as the distance between the fiber's insertion points, including wrapping around the femur, tibia, meniscus, and cartilage. This length is defined from the results of the "reference simulation" described in Cleveland State University's *Model Development* documentation.

In short, the ligament geometry was originally defined as springs that spanned between insertion points without any bone, cartilage, or meniscus wrapping. The *reference simulation* enforced ligament wrapping while maintaining the imaged joint position for the ligament insertion points, bones, cartilage, and menisci. After wrapping was enforced, the positions of the ligament nodes were extracted from the *reference simulation* results. These coordinates are used to determine each ligament fiber's insertion-to-insertion length when the joint is in the imaged position.

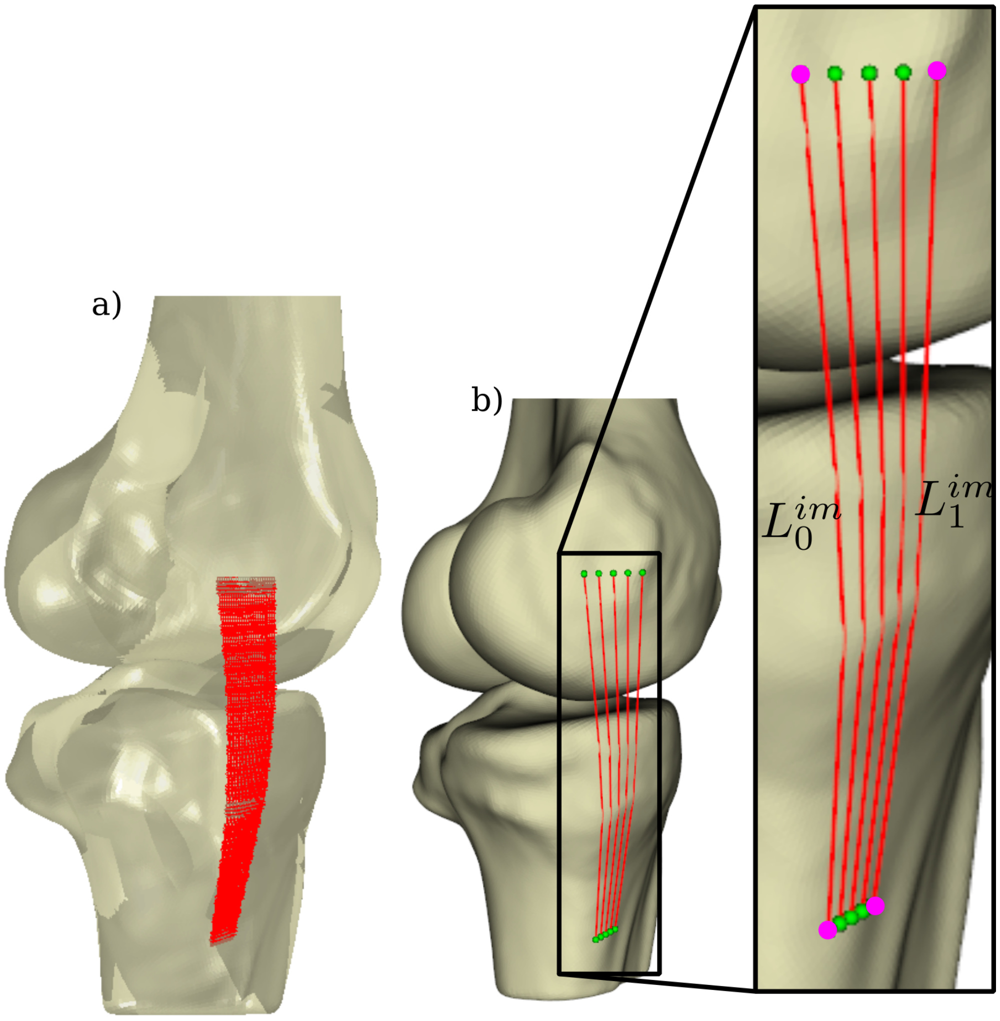


Fig. 4.1: An example of (a) a ligament mesh with 25 fibers from a finite element model. (b) A simplified representation of that same model showing 5 fibers. (inset) The magenta points show the insertions of the fibers at the margins of the ligament bundle insertion, and the insertion-to-insertion length of the first (L_0^{im}) and second (L_1^{im}) fiber at the margins of the insertion in the imaged joint position.

4.3 Control Variable Bounds

In the optimization scheme, upper and lower bounds are specified for the *control variables* (page 17) which define the ligament slack lengths. The bounds serve to avoid unstable simulations and to limit the control variable space. The upper bounds are defined from a rigid body *forward kinematic simulation* (page 19) of the *test cases* (page 16) included in the optimization scheme. The results of this analysis are used to determine the maximum length of the controlled ligament fibers, and these maximum lengths are the upper bounds of the corresponding control variable. The lower bounds are defined as a percentage of the corresponding upper bound.

4.3.1 Forward Kinematics Simulation

A rigid body forward kinematics finite element simulation is used to estimate the insertion-to-insertion length for the ligament fibers for every *test case* (page 16) that is used in the optimization scheme. The forward kinematics simulation utilizes the same tibia, femur and ligament geometries as the model used to calibrate ligament properties. Effectively, it is the same model but setup to run in full kinematics based control. In this simulation, the tibia is fixed and the femur is kinematically controlled in six degrees of freedom. Experimentally measured kinematics for each *test case* (page 16) in the optimization are applied to the femur.

Prestrains are applied to ensure that the ligaments are not slack throughout the simulated test cases. The results of each finite element simulation will be manually inspected to verify each ligament is not slack throughout the simulation. If ligaments are observed to be slack, prestrain for slack ligaments will be decreased until all ligaments are taut.

The results of the forward kinematics simulation are used to determine the ligament fiber lengths that correspond to the *control variables* (page 17). At each of the simulated test points, the node positions of the ligament fibers are used to determine the fiber's insertion-to-insertion length. The maximum insertion-to-insertion fiber length that are achieved in the simulated *test cases* (page 16) are used as the *upper bound* (page 19) for the corresponding *control variable* (page 17).

4.3.2 Upper Bounds

The upper bounds are intended to limit the size of the control variable space. A *forward kinematics* (page 19) is used simulate the *test cases* (page 16) included in the optimization, and the results of these simulations are used to determine the maximum insertion-to-insertion length of the controlled ligament fibers. These maximum insertion-to-insertion lengths are the upper bounds for the corresponding *control variables* (page 17).

4.3.3 Lower Bounds

The lower bounds are intended to prevent the optimization algorithm from evaluating unrealistically small ligament slack lengths. This serves the practical purpose of avoiding unstable simulations caused by extremely tight ligaments. The lower bound for each *control variable* (page 17) is defined as 60% of the corresponding *upper bound* (page 19).

4.4 Initial Guess

The initial guess values for the *slack lengths* (page 17) are defined using the maximum lengths of the ligaments (that was determined when defining the *upper bounds* (page 19) of the optimization variable space). The initial guess for every control variable is the corresponding upper bound (i.e. maximum length) multiplied by 0.9. This value was partially based on experience, and its purpose is to provide an initial guess that satisfies the *inequality constraints* (page 20). This initial guess is uniformly applied to every control variable in an attempt to avoid influencing the optimization solution with a judicious selection of initial guess values.

4.5 Objective Function

For each objective function evaluation, the specified *control variables* (page 17) are applied to the knee model and a subset of the experimental *test cases* (page 16) are simulated. The knee model's joint kinematics [GS83] are measured from the results of the simulation. The objective function is the sum of the squared residual between the model and experimentally measured joint kinematics.

$$\begin{aligned} f(\bar{x}) &= \sum_{j=1}^{4}\sum_{k=1}^{6} w_k\Big(M_{jk}(\bar{x}) - E_{jk}\Big)^2 \\ h(x_i) &\geq 0. \text{ for } i = 1\ldots 22 \end{aligned} \tag{4.2}$$

where $\bar{x}$ is the set of 22 control variables, x_i is the i^{th} control variable, and $h(x_i)$ is an *inequality constraint* (page 20) that is applied to every control variable, for a total of 22 inequality constraints. The experimental *test case* (page 16) is represented with j, k represents the degree of kinematic freedom, and $w_k = [0, 1, 0, 0, 2, 2]$ is the weight that is applied to each kinematic degree of freedom (Table 4.1).

Table 4.1: The weights and the corresponding kinematic degree of freedom.

i	Weight	DOF
$i = 1$	0	Medial tibial translation
$i = 2$	1	Anterior tibial translation
$i = 3$	0	Superior tibial translation
$i = 4$	0	Flexion
$i = 5$	2	Varus
$i = 6$	2	Internal tibial rotation

The weighting factor, w_k, is used to equate tibial translation to tibial rotations. A weighting factor of 2 was arbitrarily selected to equate 5^o of rotational error to 10 mm of anterior tibial translational error.

The weighting factor is also used to exclude kinematic degrees of freedom from the objective function. Flexion is excluded because the experimental flexion angles are prescribed in the finite element model. Medial and superior tibial translation are excluded from the objective function, similar to previous studies [BH96] [BCF+12] [HCA+16]. The same weights are applied to the kinematics, regardless of the experimental test that is being evaluated. This means that the same kinematic degrees of freedom are included in the objective function, regardless of the experimental load case.

4.6 Constraints

Constraints are used in the optimization scheme to enforce that every ligament experiences at least a nominal amount of force at one point the loading cycle (*test cases* (page 16)). These constraints are applied to prevent the optimization algorithm from effectively removing a ligament from the knee model. This can potentially happen if the optimization algorithm assigns an excessively long ligament slack length, preventing that ligament from having an effect on the joint's kinematics, and therefore the *objective function* (page 20).

There is one inequality constraint for every *control variable* (page 17). There is a total of 22 constraints, and each constraint is enforced using the following (4.3).

$$\begin{aligned} h(x_i) &>= 0 \\ h(x_i) &= (f_i - 0.1)x_i^4 \end{aligned} \tag{4.3}$$

where x_i is the control variable, and f_i is the maximum force that the fiber that corresponds to x_i has experienced throughout the simulated *test cases* (page 16). Notice that if f_i is less than 0.1, then $h(x_i) < 0$, and there is a violation of the constraint.

The last term in (4.3) (x_i^4) is used to provide a unique function value for $h(x_i)$. This is necessary because the value f_i can equal zero at multiple values of x_i, and without the last term, $h(x_i)$ can have the same value at multiple values of x_i. The fourth power is used to increase the order of magnitude of the constraint values.

It is also prudent to use the last term in (4.3) (x_i^4) because of the nature of the system. The *control variables* (page 17) are used to define the slack lengths of the ligament fibers. When a constraint is violated, i.e. $h(x_i) < 0$, decreasing x_i will cause $h(x_i)$ to be closer to zero. Given the behavior of the model, decreasing ligament slack lengths can increase the force that the ligament carries in the simulation.

CHAPTER

FIVE

SIMULATION STEPS

This section describes the specific steps and boundary conditions that are used during Abaqus/Explicit FE simulations of the *experimental tests* (page 16). These steps will be executed during every objective function evaluation in the *optimization* (page 16). Laxity tests from two flexion angles are included in the *calibration test cases* (page 16). To limit the simulation time, two separate simulations are run in parallel. One simulation evaluates the *calibration test cases* (page 16) at 0° flexion, and the other simulates the 90° flexion *calibration test cases* (page 16).

Though there are two separate Abaqus simulations, the same steps are applied for each simulation. The differences between the simulations are the magnitude of the specified kinematics and loads in the *Initial Orientation Step* (page 24) and *Test Case Step* (page 25) steps. The magnitude of these loads and boundary conditions are defined as `*Amplitudes` in the Abaqus .inp files, and the corresponding *processed experimental data* (page 11) is used to define the amplitudes each simulation. This section does not delineate between values that were used for each simulation.

5.1 Initial Settle Step

Before this simulation, the ligament and tendon meshes have been defined from the results of the *reference simulation* during the *Model Development* phase, so there is no overclosure between the ligaments, tendons and other bodies.

The total time for this step is 0.01 seconds.

5.1.1 Interactions

All of the desired interactions are active in this step.

Table 5.1: Active interactions between different bodies and structures during the *Initial Settle* step.

Name	Femur	Tibia	Femoral Cartilage	Medial Tibial Cartilage	Lateral Tibial Cartilage
amACL	x	x	x		
plACL	x	x	x		
alPCL	x	x	x		
pmPCL	x	x	x		
sMCLProx	x	x	x	x	
sMCLDist	x	x	x	x	
dMCL	x	x	x	x	
LCL	x	x			
ALL	x	x	x		x
PFL	x	x	x		x
OPL	x	x	x		x
Femur		x		x	x
Tibia	x		x		
Femoral Cartilage		x		x	x
Medial Tibial Cartilage	x		x		
Lateral Tibial Cartilage	x		x		

5.1.2 Kinematic Boundary Conditions

The rigid bodies are fixed in all degrees of freedom.

Femur

The femur is fixed in all degrees of freedom throughout this step.

Tibia

The tibia is fixed in all degrees of freedom throughout this step.

5.1.3 Kinetic Boundary Conditions

There are no external loads.

Femur

No external loads are applied to the femur in this step.

Tibia

No external loads are applied to the tibia in this step.

5.2 Initial Orientation Step

This step moves the femur from it's initial position to the flexion angle for the specific *test case* (page 16). The femur's flexion angle is controlled, but the femur is free to move in other degrees of freedom. The initial flexion angle is defined by the joint's position in the MR images, and the final flexion angle is defined by the flexion angle that is experimentally measured during the first simulated *test case* (page 16). There is a nominal 20 N compressive load applied to the femur throughout this step.

The total time for this step is 0.5 seconds.

5.2.1 Interactions

The orientation of the knee at the end of this step is orientation at the beginning of the first simulated *test case* (page 16). As such, all of the interactions that are desired for the simulated test are active during this step (Table 5.1).

5.2.2 Kinematic Boundary Conditions

Femur

For the simulated test, the femur was unconstrained in all directions except for rotation about the flexion axis. The angle about the femur's flexion axis was assigned throughout this step.

The initial orientation of the knee is known after the femur and tibia coordinate systems are defined with respect to the MR image's coordinate system (more details in *Femur Coordinate System* (page 8) and *Tibia Coordinate System* (page 7)). The initial flexion angle is used to define the kinematic boundary condition for the femur during this step. Rotation is applied to the connector element that corresponds to the flexion axis (see the *Model Development* documentation for details on the joint coordinate system connectors).

The final flexion angle of the knee is specified as experimentally measured flexion angle at the first simulated *test case* (page 16). Note that this is the absolute flexion angle, which is defined during *experimental data processing* (page 11) and linearly ramped from the reference state. The knee is free to move in all other degrees of freedom.

Tibia

The tibia was fixed in all degrees of freedom. This ensures that the tibia's origin is in the *desired orientation* (page 7) at the beginning of the *test case step* (page 25).

5.2.3 Kinetic Boundary Conditions

Femur

To reduce the instability in the simulation, a nominal 20 N compressive force is applied to the femur during this step. This force was applied to the connector that defines internal-external tibial rotation axis (see the *Model Development* documentation for details on the joint coordinate system connectors). No other external forces are applied to the femur.

Tibia

No external loads were applied to the tibia in this step.

5.3 Test Case Step

This step applies the experimentally measured loads to the node that is coincident with the fixed tibial coordinate system's origin. The femur is fixed throughout this step, the flexion angle is maintained, and the tibia is free to move in the other five degrees of freedom.

See the *loading profile* (page 26) section for more information on how the *calibration test cases* (page 16) are applied to the knee model in this step.

5.3.1 Interactions

All interactions that are assigned for the simulated test are active during this step (Table 5.1).

5.3.2 Kinematic Boundary Conditions

Femur

The femur is fixed in all degrees of freedom throughout this step.

Tibia

The flexion angle of the joint is specified throughout this step, however, the tibia is free to move in the other five degrees of freedom. The flexion angle is defined as the experimentally measured flexion angles. Note that this is the absolute flexion angle, defined during *experimental data processing* (page 11).

5.3.3 Kinetic Boundary Conditions

Femur

No external loads were applied to the femur in this step.

Tibia

The experimentally measured tibial loads are applied to the node that is coincident that with the fixed tibial coordinate system. A *transform* (page 7) was used to define the orientation of this node to be coincident with the fixed tibial coordinate system. The experimentally measured tibial forces are first *processed* (page 13), then used to define the profile of the tibial loads in 5 degrees of freedom. Each desired loading value will be applied as linear ramps between each loading case.

Below is an example of the loads used for a left knee specimen (such as oks003):

```
** Medial tibial drawer force, -1. for right knee and 1. for left knee
*CLOAD, amplitude=medialTibialDrawerForce, follower, op=new
JointCoordSys.1, 1, 1.
** Anterior tibial drawer force, 1. for right and left knee
*CLOAD, amplitude=anteriorTibialDrawerForce, follower, op=new
JointCoordSys.1, 2, 1.
** Distraction force, -1. for right and left knee
*CLOAD, amplitude=distractionForce, follower, op=new
JointCoordSys.1, 3, -1.
```

(continues on next page)

(continued from previous page)

```
** Varus torque, 1. for right knee and -1. for left knee
*CLOAD, amplitude=varusTorque, follower, op=new
JointCoordSys.1, 5, -1.
** Internal tibial rotation torque, 1. for right knee and -1. for left knee
*CLOAD, amplitude=internalTibialRotationTorque, follower, op=new
JointCoordSys.1, 6, -1.
```

Where each amplitude is the magnitude of the corresponding load and the descriptions of the loads match the CSU convention (described in *Kinetics Adjustment 1 - CSU convention* (page 14)). The `follower` option ensures the applied loads follow the orientation of the node `JointCoordSys.1` as the tibia moves throughout the simulation.

The node `JointCoordSys.1` is the node that is coincident with the origin of the tibia's fixed coordinate system. The integer following `JointCoordSys.1` indicates the degree of freedom that the load is applied to, and this corresponds with x_i in Table 3.1. The final integer is multiplied by the specified amplitude. This is used to adjust for a right or left knee (described in *Kinetics Adjustment 2 - Right or Left Knee* (page 14)).

Note: The loads are applied relative to this node's (`JointCoordSys.1`) coordinate system. This coordinate system is likely not coincident with the global coordinate system because a *transform* (page 7) was used to define the orientation of the node.

Tibia Loading Profile

There are a total of eight test cases that are used in model calibration. As described in the *Simulation Steps* (page 22) section, there are separate simulations for each flexion angle included in the *calibration test cases* (page 16). Four of these test cases are from the 0° flexion laxity tests, and the other four are from the 90° flexion laxity tests.

A ramp-and-hold scheme is used to apply the desired loads from the *calibration test cases* (page 16) (Fig. 5.1). The loading from each included laxity test is simulated over a step size 0.5 seconds. The load is linearly increased to the desired value at 0.3 seconds and subsequently held for 0.2 seconds (Fig. 5.1). These loading profiles apply the experimental feedback values in all 5 degrees of freedom. Note that the Fig. 5.1 highlights the dominate loading axes and every degree of freedom likely has a non-zero load throughout the simulation. Axis-specific load values will be determined during extraction of the relevant loading cases using the already developed in house *tool* (page 11).

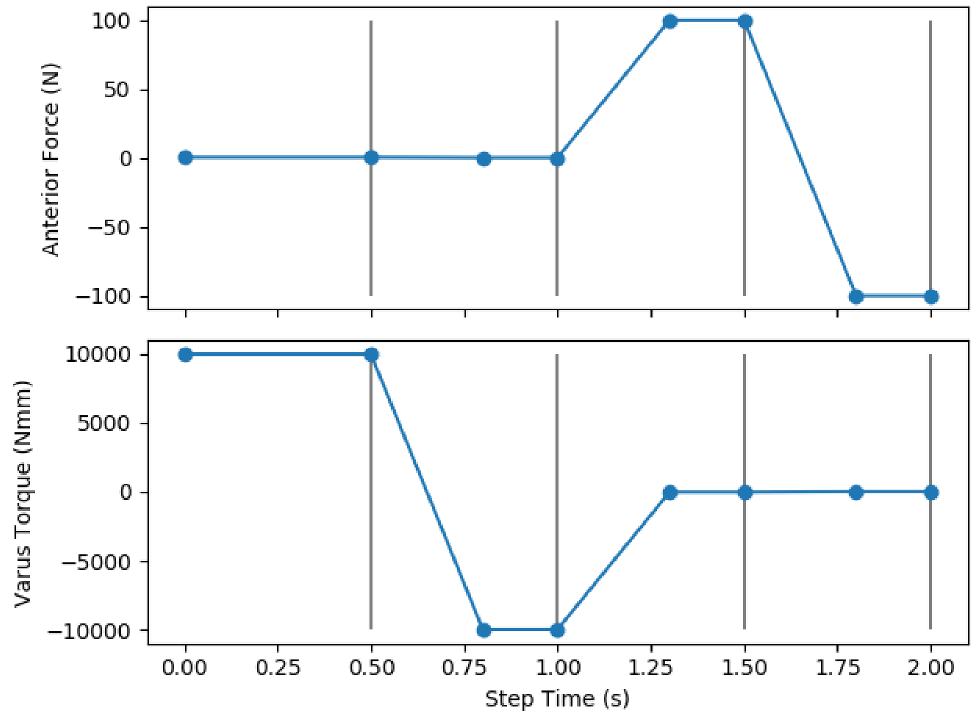


Fig. 5.1: The succession of applied loads will include the varus-valgus torques and anterior-posterior drawer loads throughout a *test case step* (page 25). The vertical lines indicate the points in the step's time where the simulation's results are extracted, which will be used in the *objective function* (page 20). This example shows the applied loads for varus torque, valgus torque, anterior-drawer and posterior drawer tests at 0.5, 1.0, 1.5, and 2.0 seconds, respectively. These points are indicated with the black vertical lines. Note that the values shown in this figure are not the explicit feedback data from OpenKnee(s), but are meant to show the approximate loads along the dominate directions for each test case.

BIBLIOGRAPHY

[EKH+15] Joseph A. Ewing, Michelle K. Kaufman, Erin E. Hutter, Jeffrey F. Granger, Matthew D. Beal, Stephen J. Piazza, and Robert A. Siston. Estimating patient-specific soft-tissue properties in a TKA knee. *Journal of Orthopaedic Research*, pages n/a–n/a, September 2015. URL: http://onlinelibrary.wiley.com/doi/10.1002/jor.23032/abstract, doi:10.1002/jor.23032.

[HCA+16] Michael D. Harris, Adam J. Cyr, Azhar A. Ali, Clare K. Fitzpatrick, Paul J. Rullkoetter, Lorin P. Maletsky, and Kevin B. Shelburne. A Combined Experimental and Computational Approach to Subject-Specific Analysis of Knee Joint Laxity. *Journal of Biomechanical Engineering*, 138(8):081004–081004, June 2016. URL: http://dx.doi.org/10.1115/1.4033882, doi:10.1115/1.4033882.

[BCF+12] Mark A. Baldwin, Chadd W. Clary, Clare K. Fitzpatrick, James S. Deacy, Lorin P. Maletsky, and Paul J. Rullkoetter. Dynamic finite element knee simulation for evaluation of knee replacement mechanics. *Journal of Biomechanics*, 45(3):474–483, February 2012. URL: http://www.sciencedirect.com/science/article/pii/S0021929011007469, doi:10.1016/j.jbiomech.2011.11.052.

[BH96] L. Blankevoort and R. Huiskes. Validation of a three-dimensional model of the knee. *Journal of Biomechanics*, 29(7):955–961, July 1996. URL: http://www.sciencedirect.com/science/article/pii/0021929095001492, doi:10.1016/0021-9290(95)00149-2.

[GS83] E. S. Grood and W. J. Suntay. A Joint Coordinate System for the Clinical Description of Three-Dimensional Motions: Application to the Knee. *Journal of Biomechanical Engineering*, 105(2):136–144, May 1983. URL: http://dx.doi.org/10.1115/1.3138397, doi:10.1115/1.3138397.

[HCA+16] Michael D. Harris, Adam J. Cyr, Azhar A. Ali, Clare K. Fitzpatrick, Paul J. Rullkoetter, Lorin P. Maletsky, and Kevin B. Shelburne. A Combined Experimental and Computational Approach to Subject-Specific Analysis of Knee Joint Laxity. *Journal of Biomechanical Engineering*, 138(8):081004–081004, June 2016. URL: http://dx.doi.org/10.1115/1.4033882, doi:10.1115/1.4033882.